Amendments to the Claims:

The below-listing of claims will replace all prior versions, and listings, of claims in the application. As presented, amendments appear in claims 1, 2, 8, 11, 12, 16, 21 and 22. All other claims remain as originally presented.

Listing of claims:

1. (Currently Amended) A method for validating executable code resident in an operating system having executable instructions, comprising the steps of:

identifying an executable code having an unaltered size;

~~receiving~~ calculating a score associated with ~~an~~ the executable code when the executable code is initially or shortly thereafter loaded into an operating system;

saving the score; ~~and~~

~~receiving~~ calculating a subsequent score on the executable code; ~~and~~

comparing the subsequent score to the saved score; ~~to determine if the executable code has been modified~~

if the subsequent score does not vary from the saved score, concluding the executable code maintains the unaltered size; and

if the subsequent score varies from the saved score, concluding the executable code has an altered size.

2. (Currently Amended) The method of claim 1, further comprising the steps of:

unloading the executable code from the operating system if the saved score is not equal to the subsequent score.

3. (Original) The method of claim 1, further comprising the steps of:

disabling at least a portion of the executable code if the saved score is not equal to the subsequent score.

4. (Original) The method of claim 1, wherein the scores are the result of a checksum calculation.

5. (Original) The method of claim 1, further comprising the steps of:

receiving one or more additional scores periodically on the executable code.

6. (Original) The method of claim 5, further comprising the steps of:

disabling at least a portion of the executable code if the saved score is not equal to any of the additional scores.

7. (Original) The method of claim 1, further comprising the steps of:

notifying electronically an owner of the executable code if the saved score is not equal to the subsequent score.

8. (Currently Amended) A method for disabling executable code which has been modified without authorization having executable instructions, comprising the steps of:

identifying an executable code having an unaltered format;

~~receiving~~ calculating a score associated with ~~an~~ the executable code;

~~receiving~~ calculating one or more subsequent scores associated with the executable code; ~~and~~

determining the executable code has an altered format if the score is not equal to any of the subsequent scores; and

disabling the executable code if the score is not equal to any of the subsequent scores.

9. (Original) The method of claim 8, further comprising the steps of:

notifying an owner of the executable code if disabled.

10. (Original) The method of claim 8, wherein the scores are the result of a checksum calculation.

11. (Currently Amended) The method of claim 8, wherein the subsequent scores are ~~received~~ calculated randomly.

12. (Currently Amended) The method of claim 8, wherein the subsequent scores are ~~received~~ calculated at one or more predetermined time intervals.

13. (Original) The method of claim 8, further comprising the steps of:

removing the executable code if disabled from a memory of an operating system wherein the executable code resides.

14. (Original) The method of claim 8, further comprising the steps of:

assisting in the loading of the executable code, if not disabled, to a memory of an operating system wherein the executable code resides.

15. (Original) The method of claim 8, further comprising the steps of:

registering the executable code if not disabled; and recording a history if the executable code is disabled.

16. (Currently Amended) A method of authenticating executable code resident in a memory having executable instructions, comprising the steps of:

identifying an executable code having an unaltered format;

acquiring a score associated with an executable code which was established when the executable code was first loaded into a memory of an operating system;

receiving a subsequent score on the executable code while the executable code is in the memory; ~~and~~

comparing the subsequent score to the score;

if the subsequent score does not vary from the score, executing the executable code in the unaltered format; and

if the subsequent score varies from the saved score, indicating the executable code has an altered format.

17. (Original) The method of claim 16, further comprising the steps of:

disabling the executable code while the executable code is in the memory when the subsequent score is not equal to the score.

18. (Original) The method of claim 16, further comprising the steps of:

suspending one or more operations of the executable code while the executable code is executing in the memory when the subsequent score is not equal to the score.

19. (Original) The method of claim 16, wherein the subsequent score is received each time the executable code is initiated in the memory for an execution.

20. (Original) The method of claim 16, reporting one or more system events and variables when the subsequent score is not equal to the score.

21. (Currently Amended) Functional data used to validate executable code embodied in a computer readable medium, the data comprising:

an unaltered original format;

a first score associated with an executable code when the executable code is initially loaded into an operating system; ~~and~~

a second score associated with the executable code at a period of time subsequent to when the executable code was initially loaded and operable to be compared with the first score to determine if the executable code has been altered since the initial load; and

an altered unoriginal format if the first and second scores do not equal.

22. (Currently Amended) A system for validating executable code, comprising:

an executable code having an unaltered size;

a scoring set of executable instruction operable to receive and record a score associated with ~~an~~ the executable code when the code is initially loaded into a computer readable medium; ~~and~~

a comparing set of executable instructions operable to receive a subsequent score associated with the code and to compare the score and the subsequent score to determine if the code has been altered; and

an executable code having an altered size if the score and the subsequent score do not equal.